PRML-PJ GNN 模型方法简述

1 .运行方式：

在main.py 函数最后一行调用process\_fold函数处修改想要选择的模型(“gcn”, ”gat”, “gin”, “sage”,“dense\_gcn”, “dense\_sage”, “dense\_cheb”)。在命令行执行python main.py或者使用PyCharm运行main.py即可。需预装torch, dgl, rdkit, sklearn等库。

2 .提交文件及输入/输出介绍

提交\*.py代码部分包含main, model, dense\_model, train, evaluate五个\*.py文件。main.py中load\_data函数为读入fold\_x文件夹中三个\*.csv文件，process\_fold函数为执行10个fold\_x,对于每个fold in range(10)，先用load\_data函数读取train, dev, test三组数据，并将smiles字符串转化为DGL图，与对应的activity值一起转化为程序调用中的data数据。sampler为通过Sampler和Randomsampler从train\_data数据中选取的部分符合设定activity=1比例的数据集，并将其作为训练数据集，利用train\_model函数对每个fold最多进行max\_iteration次训练，每次训练抽取batch\_per\_epoch组数据，累加所有随机抽取的数据组的loss，由此对模型优化，训练的结果利用evaluate\_model 函数对测试集进行测试，得到ROC-AUC / PRC-AUC。model与dense\_model里分别包含GIN, GCN, GAT, SAGE, DenseGraph, DenseSAGE, DenseCheb两种六类模型，两种model的区别在于图构建（process函数）方式不同。GNN模型中包含GNN模型初始化，构建smiles字符串对应化学式的DGL图和每个原子的特征向量（利用atom\_feature函数），forward以及predict部分。其中forward部分对每组batch中的数据的图邻接矩阵利用预先构建的 GNN 模块获得标准的两层GNN模型，获得的结果用criterion（CrossEntropyLoss）预测并统计loss。

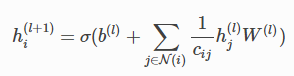
输入为main.py目录下train\_cv中10个fold，输出为main.py目录下output.txt文件。输出文件每个cycle在输出文件中显示loss, tn, tp, fn, fp，每个fold训练结束后对测试集评估，并输出ROC-AUC和PRC-AUC的值，之后为对十组ROC-AUC/PRC-AUC的分析。输出文件最后统计每个fold\_x运行的时间和程序运行总时间。

3 .模型介绍

model.py, models.py中分别包含Conv Layers:(gcn (GraphConv), gat (GATConv), gin (GINConv), sage (SAGEConv))和Dense Conv Layers: (dense\_gcn (DenseGraphConv), dense\_sage (DenseSAGEConv), dense\_cheb (DenseChebConv) )两种七个模型。

3 .1 GraphConv(GCN)：

**· Classdgl.nn.pytorch.conv.GraphConv(in\_feats, out\_feats, norm='both', weight=True, bias=True, activation=None)**

可以描述为：，其中1593692908(1)是节点i的相邻节点集，cij=1593692962(1)，σ是激活函数。权重W(l)使用Glorot均匀初始化进行初始化，并且偏差被初始化为零。

**· forward(graph, feat, weight=None)**

3 .2 GATConv：

**· classdgl.nn.pytorch.conv.GATConv(in\_feats, out\_feats, num\_heads, feat\_drop=0.0, attn\_drop=0.0, negative\_slope=0.2, residual=False, activation=None)**

公式描述：1593693339(1)，其中αij是结点i和j之间的attention score，1593693431(1)，1593693447(1)。

**·forward(graph, feat)**

3 .3 GINConv：

**·classdgl.nn.pytorch.conv.GINConv(apply\_func, aggregator\_type, init\_eps=0, learn\_eps=False)**

公式表达：1593693509(1)

其中aggregator\_type = sum, max, mean

**·forward(graph, feat)**

3 .4 SAGEConv：

**· classdgl.nn.pytorch.conv.SAGEConv(in\_feats, out\_feats, aggregator\_type, feat\_drop=0.0, bias=True, norm=None, activation=None)**

公式表达：1593693706(1)，1593693719(1)，1593693737(1)，其中****aggregator\_type = mean, gcn, pool, lstm****

****·**forward(graph, feat)**

3 .5 DenseGraphConv：

**· classdgl.nn.pytorch.conv.DenseGraphConv(in\_feats, out\_feats, norm='both', bias=True, activation=None)**

图卷积网络层的图结构由邻接矩阵给出。

·**forward(adj, feat)**

3 .6 DenseSAGEConv：

**· classdgl.nn.pytorch.conv.DenseSAGEConv(in\_feats, out\_feats, feat\_drop=0.0, bias=True, norm=None, activation=None)**

图卷积网络层的图结构由邻接矩阵给出。

·**forward(adj, feat)**

3 .7 DenseChebConv：

**·classdgl.nn.pytorch.conv.DenseChebConv(in\_feats, out\_feats, k, bias=True)**

·**forward(adj, feat, lambda\_max=None)**

3 .8 Conv Layers与Dense Conv Layers的区别在于forward时第一个参量是DGL图还是应用图卷积的图的邻接矩阵。在构建模型时两者(model.py中class Model与models.py中class Models)的区别在于对smiles字符串进行处理的process函数，其中一个是构图另一个是构建图卷积的邻接矩阵。dgl.nn.pytorch中包含多种神经网络模型，此次试验只是选取其中七个函数进行运行比较。

4 .模型运行结果与时间表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| model fold | GCN | | | | | GAT | | | |
| ROC-AUC | PRC-AUC | | Runtime(s) | | ROC-AUC | PRC-AUC | | Runtime(s) |
| 0 | 0.791 | 0.354 | | 533 | | 0.630 | 0.184 | | 860 |
| 1 | 0.891 | 0.406 | | 725 | | 0.910 | 0.521 | | 2180 |
| 2 | 0.792 | 0.215 | | 356 | | 0.660 | 0.219 | | 2501 |
| 3 | 0.521 | 0.022 | | 746 | | 0.403 | 0.033 | | 2519 |
| 4 | 0.863 | 0.711 | | 790 | | 0.921 | 0.468 | | 1594 |
| 5 | 0.542 | 0.343 | | 234 | | 0.584 | 0.344 | | 2624 |
| 6 | 0.427 | 0.020 | | 777 | | 0.845 | 0.044 | | 2475 |
| 7 | 0.454 | 0.004 | | 644 | | 0.884 | 0.020 | | 2534 |
| 8 | 0.981 | 0.513 | | 661 | | 0.891 | 0.760 | | 2579 |
| 9 | 0.882 | 0.369 | | 508 | | 0.948 | 0.118 | | 4142 |
| TOTAL | 0.715±0.195 | 0.296±0.22 | | 5974 | | 0.767±0.176 | 0.271±0.235 | | 24008 |
| model fold | GIN | | | | | SAGE | | | |
| ROC-AUC | PRC-AUC | | Runtime(s) | | ROC-AUC | | PRC-AUC | Runtime(s) |
| 0 | 0.790 | 0.479 | | 456 | | 0.791 | | 0.311 | 683 |
| 1 | 0.927 | 0.635 | | 537 | | 0.804 | | 0.412 | 750 |
| 2 | 0.766 | 0.197 | | 539 | | 0.727 | | 0.524 | 495 |
| 3 | 0.657 | 0.103 | | 539 | | 0.648 | | 0.060 | 446 |
| 4 | 0.857 | 0.728 | | 535 | | 0.938 | | 0.735 | 765 |
| 5 | 0.505 | 0.342 | | 578 | | 0.611 | | 0.351 | 751 |
| 6 | 0.714 | 0.025 | | 538 | | 0.604 | | 0.075 | 717 |
| 7 | 0.908 | 0.025 | | 560 | | 0.826 | | 0.014 | 506 |
| 8 | 0.978 | 0.497 | | 562 | | 0.766 | | 0.431 | 1880 |
| 9 | 0.861 | 0.073 | | 560 | | 0.888 | | 0.095 | 1152 |
| TOTAL | 0.796±0.135 | 0.310±0.249 | | 5403 | | 0.760±0.107 | | 0.301±0.225 | 8144 |
| model fold | DenseGCN | | | | | DenseSAGE | | | |
| ROC-AUC | PRC-AUC | | Runtime(s) | | ROC-AUC | | PRC-AUC | Runtime(s) |
| 0 | 0.702 | 0.157 | | 133 | | 0.541 | | 0.267 | 71 |
| 1 | 0.918 | 0.707 | | 90 | | 0.916 | | 0.676 | 80 |
| 2 | 0.835 | 0.515 | | 84 | | 0.868 | | 0.282 | 164 |
| 3 | 0.718 | 0.274 | | 90 | | 0.739 | | 0.276 | 87 |
| 4 | 0.936 | 0.689 | | 222 | | 0.939 | | 0.698 | 221 |
| 5 | 0.877 | 0.289 | | 86 | | 0.736 | | 0.672 | 90 |
| 6 | 0.210 | 0.008 | | 223 | | 0.279 | | 0.009 | 218 |
| 7 | 0.106 | 0.003 | | 234 | | 0.121 | | 0.003 | 229 |
| 8 | 0.981 | 0.531 | | 228 | | 0.997 | | 0.908 | 207 |
| 9 | 0.812 | 0.370 | | 90 | | 0.853 | | 0.069 | 85 |
| TOTAL | 0.709±0.289 | 0.354±0.242 | | 1479 | | 0.699±0.280 | | 0.386±0.310 | 1451 |
| model fold | DenseCheb | | | | |
| ROC-AUC | | PRC-AUC | | Runtime(s) |
| 0 | 0.614 | | 0.183 | | 469 |
| 1 | 0.915 | | 0.745 | | 438 |
| 2 | 0.701 | | 0.512 | | 444 |
| 3 | 0.793 | | 0.059 | | 418 |
| 4 | 0.950 | | 0.848 | | 409 |
| 5 | 0.781 | | 0.392 | | 362 |
| 6 | 0.290 | | 0.009 | | 312 |
| 7 | 0.319 | | 0.004 | | 326 |
| 8 | 0.989 | | 0.656 | | 324 |
| 9 | 0.645 | | 0.019 | | 328 |
| TOTAL | 0.700±0.230 | | 0.343±0.313 | | 3829 |

5 .模型运行效率和运行结果比较

·运行时间比较：DenseSAGE < DenseGraph < DenseCheb < GCN ≈ GIN< SAGE << GAT

· ROC-AUC 数值比较：GIN优于GAT, SAGE优于DenseSAGE优于GCN,DenseGraph,DenseCheb，误差GCN < DenseCheb < DenseGraph

· PRC-AUC数值比较：DenseSAGE优于DenseCheb,DenseGraph优于GIN, SAGE, GCN优于GAT，误差DenseGraph < DenseCheb

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| model fold | GIN(aggregator\_type=sum) | | | GIN(aggregator\_type=mean) | | | GIN(aggregator\_type=max) | | |
| ROC-AUC | PRC-AUC | Runtime(s) | ROC-AUC | PRC-AUC | Runtime(s) | ROC-AUC | PRC-AUC | Runtime(s) |
| 0 | 0.790 | 0.479 | 456 | 0.655 | 0.083 | 256 | 0.843 | 0.210 | 801 |
| 1 | 0.927 | 0.635 | 537 | 0.855 | 0.581 | 631 | 0.871 | 0.429 | 659 |
| 2 | 0.766 | 0.197 | 539 | 0.717 | 0.331 | 452 | 0.772 | 0.334 | 1866 |
| 3 | 0.657 | 0.103 | 539 | 0.605 | 0.082 | 517 | 0.778 | 0.065 | 1844 |
| 4 | 0.857 | 0.728 | 535 | 0.926 | 0.279 | 334 | 0.978 | 0.833 | 849 |
| 5 | 0.505 | 0.342 | 578 | 0.562 | 0.345 | 242 | 0.666 | 0.381 | 809 |
| 6 | 0.714 | 0.025 | 538 | 0.601 | 0.055 | 683 | 0.708 | 0.045 | 802 |
| 7 | 0.908 | 0.025 | 560 | 0.507 | 0.005 | 686 | 0.758 | 0.010 | 355 |
| 8 | 0.978 | 0.497 | 562 | 0.789 | 0.643 | 646 | 0.912 | 0.403 | 827 |
| 9 | 0.861 | 0.073 | 560 | 0.846 | 0.067 | 645 | 0.928 | 0.110 | 802 |
| TOTAL | 0.796±0.135 | 0.310±0.249 | 5403 | 0.706±0.135 | 0.247±0.216 | 5091 | 0.821±0.096 | 0.282± 0.237 | 9613 |

·整体比较来看，Conv Layers计算的ROC-AUC 和 PRC-AUC的值比Dense Conv Layers更高，且误差较小，但用时更长。

GINConv模型取aggregator\_type=mean最优

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| model fold | SAGE(aggregator\_type=mean) | | | SAGE(aggregator\_type=gcn) | | | SAGE(aggregator\_type=pool) | | |
| ROC-AUC | PRC-AUC | Runtime(s) | ROC-AUC | PRC-AUC | Runtime(s) | ROC-AUC | PRC-AUC | Runtime(s) |
| 0 | 0.791 | 0.311 | 683 | 0.716 | 0.345 | 318 | 0.796 | 0.355 | 789 |
| 1 | 0.804 | 0.412 | 750 | 0.840 | 0.309 | 222 | 0.921 | 0.740 | 1673 |
| 2 | 0.727 | 0.524 | 495 | 0.736 | 0.148 | 287 | 0.662 | 0.511 | 2141 |
| 3 | 0.648 | 0.060 | 446 | 0.601 | 0.026 | 366 | 0.630 | 0.038 | 1466 |
| 4 | 0.938 | 0.735 | 765 | 0.884 | 0.501 | 1583 | 0.886 | 0.687 | 956 |
| 5 | 0.611 | 0.351 | 751 | 0.640 | 0.347 | 870 | 0.599 | 0.071 | 930 |
| 6 | 0.604 | 0.075 | 717 | 0.578 | 0.075 | 1548 | 0.516 | 0.014 | 1991 |
| 7 | 0.826 | 0.014 | 506 | 0.667 | 0.007 | 552 | 0.696 | 0.008 | 1385 |
| 8 | 0.766 | 0.431 | 1880 | 0.949 | 0.406 | 940 | 0.990 | 0.729 | 839 |
| 9 | 0.888 | 0.095 | 1152 | 0.943 | 0.402 | 309 | 0.826 | 0.053 | 424 |
| TOTAL | 0.760±0.107 | 0.301±0.225 | 8144 | 0.755±0.132 | 0.256±0.168 | 6993 | 0.752±0.147 | 0.321±0.303 | 12596 |

SAGEConv模型取aggregator\_type=gcn最优

Reference

[1] https://blog.csdn.net/qq\_27075943/article/details/106623059 #【GNN】MPNN：消息传递神经网络

[2] https://blog.csdn.net/Frank\_LJiang/article/details/95194733 #图神经网络（GNN）总结（GCN）

[3] https://zhuanlan.zhihu.com/p/89503068 #一文读懂图卷积GCN

[4] http://rdkit.chenzhaoqiang.com/basicManual.html #RDKit中文教程

[5] https://blog.csdn.net/qq\_33148001/article/details/105192816 #DGL官方教程--API--dgl.nn